



Certificación PCAP

Lección 1: Presentación e introducción a la certificación PCAP

ÍNDICE

Lección 1. – Presentación e introducción a la certificación PCAP	2
Presentación y objetivos	2
1. Qué es python	3
2. Por qué python.....	6
3. Open Education and Development Group (OpenEDG) y otras instituciones	11
4. PCAP. Python Institute y otras certificaciones	14
5. Índice de contenidos PCAP	16
5.1 Temario PCAP-31-02 (PVTs).....	16
5.2 Temario PCAP-31-03 (PVTs).....	19
6. Recursos. OpenEDG y Cisco Networking Academy.....	22
7. Objetivos de los cursos	24
8. Examen PCAP.....	25
8.1 Precio del PCAP y descuentos disponibles	25
9. Acreditación digital	27
10. Bibliografía	28

Lección 1. – Presentación e introducción a la certificación PCAP

PRESENTACIÓN Y OBJETIVOS

El objetivo principal de esta primera lección es la presentación de la asignatura. Hablaremos brevemente de las características principales de Python, así como, la gran importancia que tiene hoy en día en el ámbito tecnológico e intentaremos justificar objetivamente por qué. Así mismo, realizaremos la introducción a la certificación PCAP dentro del conjunto de certificaciones en el lenguaje de programación Python ofrecidas por “Python Institute”.

Con la intención de tener una perspectiva panorámica, hablaremos brevemente sobre el “ecosistema” de instituciones que entran en juego y hacen posible la existencia tanto de las certificaciones en Python (y otras) como de los recursos de aprendizaje que estas ponen a nuestra disposición. Introduciremos, también, el índice de contenidos de la certificación y sus modificaciones en comparación a la nueva versión del examen.

Posteriormente, presentaremos los cursos ofrecidos por OpenEDG que utilizaremos como apoyo principal en esta asignatura, ya que estos están específicamente preparados y orientados al estudio del índice de contenidos del PCAP y consecuentemente serán nuestros mejores aliados si nos planteamos obtener dicha certificación.

Finalmente, y no por ello menos importante, hablaremos del examen y de su costo del examen y también de las oportunidades de descuento que podemos obtener para su realización.

1. QUÉ ES PYTHON

En primer lugar, es importante conocer qué es Python y sus características principales. Python es un lenguaje de programación de **alto nivel, interpretado, orientado a objetos** y de uso generalizado que se utiliza para la **programación de propósito general**.

Veamos esto un poco más en detalle. ¿Qué significa que se trata de un lenguaje de alto nivel? Pues significa que se trata de un lenguaje cercano al lenguaje natural pero mucho más sencillo que éste y, a la vez, mucho más complejo que el lenguaje máquina o los lenguajes de bajo nivel (ensamblador). Es decir, un lenguaje de alto nivel es similar al lenguaje natural ya que usa símbolos, palabras y convenciones legibles para los humanos y nos permite expresar comandos a computadoras que son mucho más complejas que la lista de instrucciones (IL) que estas entienden.

Es importante conocer que hay dos formas diferentes de transformar un programa de un lenguaje de programación de alto nivel a un lenguaje de máquina:

- | **Compilación** - El programa fuente se traduce una vez obteniendo un archivo que contiene el código máquina (por ejemplo, un .exe si se trabaja en windows) y es este el archivo que se distribuye. Cada vez que se modifica el código fuente debe repetirse esta traducción. El programa que realiza la traducción se llama compilador o traductor.
- | **Interpretación** - El programa fuente se traduce a código máquina cada vez que se ejecuta el programa. El programa que realiza este tipo de transformación se denomina intérprete ya que interpreta el código cada vez que se ejecuta. El usuario final necesitará el código fuente más el intérprete para poderlo ejecutar.

	COMPILACIÓN	INTERPRETACIÓN
V E N T A J A S	<ul style="list-style-type: none"> La ejecución del código suele ser más rápida. Sólo el usuario programador debe tener el compilador, el usuario final puede ejecutar el programa sin él. El código se distribuye en lenguaje máquina, por lo que es probable que los propios inventos o trucos de programación sigan siendo secreto. 	<ul style="list-style-type: none"> Puedes ejecutar el código en cuanto lo modifiques o completes, sin tener que esperar a compilarlo El código se almacena y distribuye en el lenguaje de programación; esto significa que puede ejecutarse en máquinas con diferente lenguaje máquina (multiplataforma).
I N C O N V E N I E N T E S	<ul style="list-style-type: none"> La compilación puede llevar mucho tiempo y después de cada modificación es necesario compilar el código. Necesitas tener tantos compiladores como plataformas hardware en los que deseas que se ejecute tu código. 	<ul style="list-style-type: none"> Tu código comparte la potencia de la máquina donde se ejecuta con el intérprete, por lo que puede que no sea tan rápido. Tanto el programador como el usuario final necesitan tener instalado en su máquina el intérprete.

Python es un lenguaje interpretado, es decir, no es necesario compilarlo para ejecutarlo, sino que se ejecuta directamente en el ordenador utilizando el **intérprete**, heredando todas las ventajas de los lenguajes interpretados:

- Permite ejecutar el código en cuanto se termina de escribir ya que no hay fases adicionales de traducción. En cambio, en los lenguajes compilados una vez se termina de escribir o modificar el código es necesario compilarlo lo que puede llevar mucho tiempo.
- El código se almacena utilizando el lenguaje de programación, no el de la máquina; esto significa que puede ejecutarse en máquinas que utilizan diferentes lenguajes máquina. Es decir, es **multiplataforma**. En cambio, en los lenguajes compilados el código se guarda en lenguaje máquina por lo que si se quiere utilizar en otra máquina con un lenguaje máquina diferente es necesario compilarlo de nuevo para esa máquina.

Además, Python es un lenguaje de programación de propósito general, esto quiere decir que puede ser usado para casi todo como veremos más adelante.

Por último, es importante conocer que **Python 3** es la versión más actual y es la que se utilizará durante la asignatura. Python 3 no es compatible con la versión anterior, Python 2, ya que Python 3 no es solo una versión mejorada de Python2, sino que es un lenguaje completamente diferente, y aunque son muy similares existen pequeñas diferencias significativas que hacen que no sean compatibles.

2. POR QUÉ PYTHON

Resulta verdaderamente impresionante observar los increíbles avances que han tenido lugar en el ámbito tecnológico en las dos últimas décadas y los que actualmente continuamos observando. De la misma manera, con este avance, la “oferta” de lenguajes de programación que podemos aspirar a dominar o en los que podemos decidir enfocarnos para poder labrar nuestro futuro profesional es también enorme. Haciendo una breve investigación (en google, por supuesto ;)) podemos ver que, a día de hoy, esta “oferta” a la que hacemos referencia cuenta con casi 700[1] lenguajes de programación distintos. Si bien es cierto, muchos de ellos van decayendo con el tiempo y se van haciendo menos populares, hasta el punto de *casi* desaparecer cuando su “época” termina. Podría ser el ejemplo de Basic, Delphi, Pascal o Objective-C. Entonces, siendo esto así, ¿cómo podríamos estar seguros de que el lenguaje de programación que decidimos estudiar (en este caso Python) no se encuentra entre los “condenados a desaparecer” y estamos en realidad haciendo una mala inversión con nuestro tiempo y recursos?

Pues la respuesta es bastante sencilla, Python no se acerca ni de lejos a ser un lenguaje de programación en “vías de extinción” ni su popularidad está decayendo, más bien todo lo contrario, desde que fue creado por Guido van Rossum en 1991[2], su popularidad no ha dejado de crecer, así como las áreas tecnológicas en las que su uso es relevante. Aun así, esta afirmación podría ser solamente una impresión personal, subjetiva y/o parcial de lo que ocurre en realidad en el mundo hoy en día en relación a la popularidad de los distintos lenguajes de programación. Es por ello por lo que a continuación, haremos referencia a los resultados obtenidos tanto por TIOBE como los obtenidos por PYPL (Popularity of Programming Language). Estas plataformas, analizan la evolución de la popularidad de los lenguajes de programación en el tiempo de una manera “sencilla” utilizando el número de búsquedas realizadas conteniendo el nombre de cada uno de los lenguajes de programación estudiados. A continuación, podemos ver algunos de los resultados obtenidos por ambos.











Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	▲	 Python	13.58%	+1.86%
2	1	▼	 C	12.44%	-4.94%
3	2	▼	 Java	10.66%	-1.30%
4	4		 C++	8.29%	+0.73%
5	5		 C#	5.68%	+1.73%
6	6		 Visual Basic	4.74%	+0.90%
7	7		 JavaScript	2.09%	-0.11%
8	11	▲	 Assembly language	1.85%	+0.21%
9	12	▲	 SQL	1.80%	+0.19%
10	13	▲	 Swift	1.41%	-0.02%

Figura 2.1. Tabla comparativa de índice de popularidad de lenguajes de programación de TIOBE entre 2021 y 2022[3]

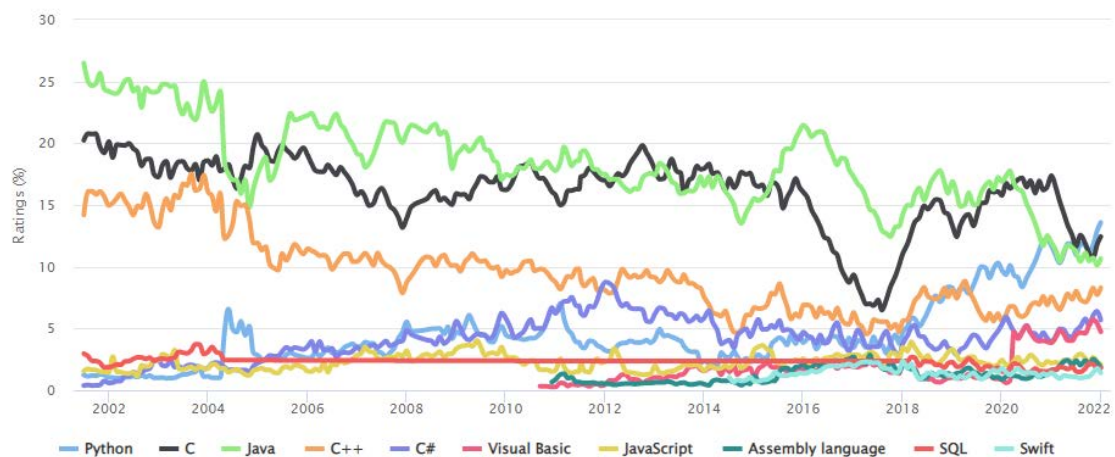


Figura 2.2. Representación gráfica de la evolución del índice de popularidad para distintos lenguajes de programación entre los años 2002 y 2022 por TIOBE[3]

En los resultados de TIOBE, podemos observar como Python se encuentra ocupando la primera posición en el ranking, por delante de los gigantes de C y Java. Además, según las tendencias que se pueden observar en ambas figuras, Python continúa al alza mientras que Java y C tienen una pequeña tendencia negativa.

Por otro lado, también podemos traer a colación los resultados obtenidos por PYPL, igualmente válidos para el apoyo de la teoría de que Python no es ni mucho menos un lenguaje poco popular hoy en día.

Worldwide, Jan 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %
6		PHP	6.06 %	+0.0 %
7		R	4.19 %	+0.3 %
8		Objective-C	2.27 %	-1.4 %
9		Swift	1.91 %	-0.2 %
10		TypeScript	1.74 %	-0.0 %

Figura 2.3. Tabla comparativa de índice de popularidad de lenguajes de programación de PYPL entre 2021 y 2022[4]

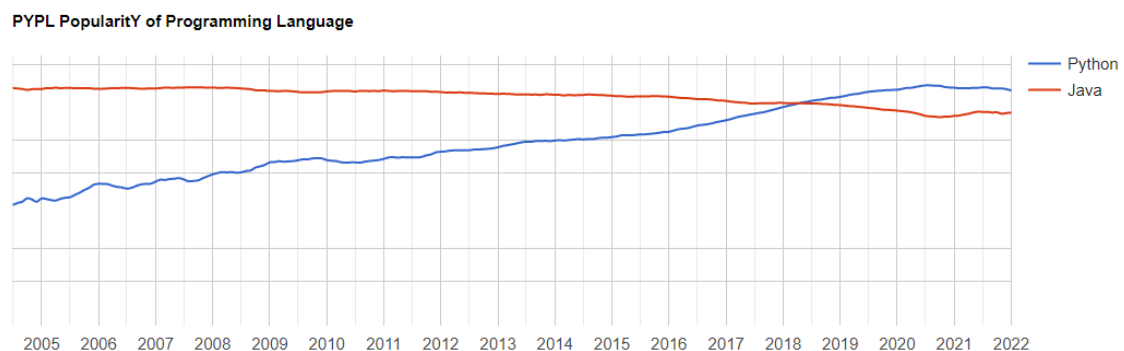


Figura 2.4 . Representación gráfica comparativa de la evolución del índice de popularidad entre Java y Python por PYPL[4]

Como podemos observar, los resultados obtenidos por TIOBE y PYPL no son idénticos, esto es debido a que tampoco lo son los criterios establecidos por cada uno para obtener su índice de popularidad. Podemos encontrar una detallada definición del índice de TIOBE en su web[5], donde, entre otros factores, encontraremos que utilizan los datos provistos por los mayores motores de búsqueda (*Google, Yahoo, Youtube, Wikipedia, entre otros*) para nutrir su algoritmo y realizar el cálculo del índice.

Por su parte, PYPL utiliza datos obtenidos de "*Google Trends*" para contabilizar el número de búsquedas en relación a cada lenguaje de programación y es de esta manera como calculan la popularidad y tendencias de los mismos.

Sea como fuere, tanto los resultados obtenidos por TIOBE como los obtenidos por PYPL nos confirman que Python tiene una relevancia muy significativa en el mundo tecnológico hoy en día, y que su popularidad parece no haber alcanzado su pico máximo.

Ahora podríamos pensar, - vale, Python es "guay" dentro del mundo tecnológico, pero, ¿qué significa que sea popular? ¿es simplemente una moda pasajera? o ¿existe alguna razón detrás de esta popularidad?

Por supuesto que hay una razón, y no solo una, para que Python sea tan utilizado. Algunas de ellas son:

- | Es un lenguaje **fácil de aprender**. En comparación a otros lenguajes como podrían ser Java o C, Python tiene una sintaxis muy corta y muy entendible. Es también poco verboso lo cual lo hace mucho más asequible en cuanto a dificultad ya que tiene una curva de aprendizaje muy suave.
- | Es **fácil de usar**. Al ser poco verboso, desarrollar en Python puede ser, a su vez, más rápido que con otros lenguajes.
- | Es **fácil de entender**. La simplicidad, poca verbosidad y sintaxis "ordenada" de Python lo convierten en un lenguaje fácil de entender, es decir, requiere menos esfuerzo poder entender el código desarrollado por otros.
- | Es **fácil de obtener y de desplegar**. Esta es una de las mayores ventajas de Python, es **código abierto, gratis y multiplataforma**, lo cual lo hace muy atractivo para su extenso uso en multitud de disciplinas.

| Por último, pero no por ello menos importante, Python está respaldado por una **gran comunidad**. Puede resultar obvio, y por ello podría pasarsenos por alto, pero el hecho de que haya muchas personas utilizando y desarrollando Python hacen que sea relativamente fácil poder encontrar librerías para casi cualquier finalidad que necesitemos y que igualmente sea sencillo encontrar soluciones a la mayoría de problemas que podemos encontrar al utilizarlo mediante el apoyo de esta extensa comunidad.

Como vemos, las ventajas de utilizar python no son pocas y tampoco lo son las numerosas áreas tecnológicas en las que podemos encontrarlo. Desde desarrollo web con librerías como *Django*, *Flask* o *Pyramid* hasta el famoso y omnipresente *Machine Learning* o *Inteligencia Artificial* con librerías como *PyTorch* y *Scikit-learn*. Por supuesto, también podemos encontrar a Python en el campo de *Data science* (*Pandas*, *Numpy*, *Matplotlib*, *Scipy*, etc) y *Big data* (*PySpark* y *Dask*), o en la administración de sistemas y DevOps en herramientas como *Ansible*. Python tiene, así mismo, una gran relevancia dentro del área de la ciberseguridad, ingeniería de redes y en el creciente mundo del *Internet of Things* (IoT).

Por supuesto, estos no son los únicos campos en los que podemos ver a python en acción, pero quizás sean los más relevantes y actuales.

Una vez analizada y confirmada la “popularidad” de este lenguaje y entendidos los motivos que se encuentran detrás de esta, ahora sí que podemos estar seguros de que dedicar tiempo y recursos a aprender y/o certificarnos en Python, es una muy buena inversión para nuestras carreras profesionales que tarde o temprano dará su merecida recompensa.

3. OPEN EDUCATION AND DEVELOPMENT GROUP (OPENEDG) Y OTRAS INSTITUCIONES

Como es lógico, la posibilidad de obtener un título o certificado que acredite a nivel global que tenemos unos conocimientos y experiencia en determinado ámbito del conocimiento no sería posible sin la existencia de una o varias instituciones que se encargaran de respaldarlos. Concretamente, sería necesario, primeramente, una organización que promoviera activamente el estudio de las disciplinas de interés. Por supuesto, también sería necesaria la existencia de algún tipo de formación o cursos para los interesados, junto con los procesos de examinación adecuados, entendiéndose por “adecuados”, procesos lo suficientemente rigurosos como para que la certificación, una vez obtenida no carezca de valor y pueda en verdad ser un título “global” y reconocido en cualquier parte del mundo. Por lo tanto, preferentemente, esta examinación ha de ser llevada a cabo por una institución externa a la que realiza la formación.

Si nos estamos planteando la posibilidad de obtener la certificación PCAP, debería interesarnos, por lo tanto, saber qué plataformas tenemos a nuestro alcance para formarnos, así como cuál es el proceso de examinación, dónde tiene lugar y/o qué institución lo realiza y todos los detalles en relación a esto. Por ello, conocer de manera general las instituciones que se encuentran detrás de las certificaciones en Python y saber qué papel juega cada una de ellas puede resultarnos verdaderamente útil.

Open Education and Development Group (De aquí en adelante OpenEDG) (<https://openedg.org>)

Esta organización fue creada en 2011 con el objetivo principal de promover un programa de excelencia en el ámbito del desarrollo y las tecnologías de la información.

En definitiva, es una plataforma educativa que promueve la idea de una sociedad moderna y “digitalmente educada” y para ello han ido trabajando en el establecimiento de los estándares de conocimiento en las disciplinas en las que centran su formación a la vez que proveen oportunidades a aquellos que desean comenzar o continuar una carrera en el ámbito de las nuevas tecnologías y la programación.

El pilar fundamental en el que se apoyan es, como se puede imaginar, la buena educación, quedando de manifiesto en la numerosa oferta de cursos gratuitos que ponen al alcance de cualquiera a través de su “herramienta de aprendizaje interactivo” llamada **Edube Interactive**.

OpenEDG se centra principalmente en los lenguajes de programación de Python, C y C++, poniendo a nuestro alcance cursos desde nivel básico a avanzado de cada uno de ellos.



Edube Interactive™

Python Institute (<https://pythoninstitute.org>)

Por otro lado, tenemos a Python Institute. Esta institución procede de OpenEDG y cómo podemos claramente intuir por su nombre, esta institución se encarga de todo lo que tiene que ver con Python, sus distintas certificaciones, y el establecimiento del estándar en esta rama del conocimiento. Hablaremos en los siguientes puntos con un poco más de detalle ella, puesto que es de la que parte directamente la certificación PCAP.



Cisco Networking Academy (<https://www.netacad.com>)

Paralelamente a OpenEDG y Python Institute, tenemos a Cisco Networking Academy colaborador directo pero independiente de estas últimas. Esta institución, al igual que OpenEDG, se encarga principalmente de promover la educación digital y apoyar a aquellos que persigan una carrera en el ámbito de las nuevas tecnologías. Igualmente, ponen a disposición de los interesados valiosos recursos educativos (algunos sin costo, otros de pago).

Como diferencia destacable, podríamos decir que Cisco Networking Academy no solo ofrece formación relacionada con los lenguajes de programación Python, C y C++, sino que su oferta de formación es un poco más amplia, incluyendo en su currículum desde formación en el ámbito de las redes y comunicaciones, ciberseguridad hasta Internet of Things, automatización de infraestructuras, programación o formación en manejo de sistemas operativos como Linux.



Pearson VUE (<https://home.pearsonvue.com>)

Finalmente, tenemos a Pearson VUE. Esta institución se encarga de otorgar valor y reconocimiento global a las certificaciones mediante la elaboración de procesos de evaluación que cumplan con el estándar de calidad adecuado. En otras palabras, se encargan de diseñar los exámenes a los que los candidatos deberán enfrentarse para la obtención de las certificaciones, así como del proceso mediante el cual se llevan a cabo. Pearson VUE tiene presencia en más de 180 países y cada año realizan más de 15 millones de exámenes en 47 idiomas distintos. Hablaremos más en detalle posteriormente sobre este proceso y los lugares y formatos que existen para poder realizar los exámenes.



4. PCAP. PYTHON INSTITUTE Y OTRAS CERTIFICACIONES

Hablaremos brevemente de Python Institute, quiénes son, qué certificaciones adicionales ponen a nuestro alcance y qué papel tiene la certificación PCAP dentro del currículum de certificaciones en Python.

Python Institute es una organización sin ánimo de lucro que se encarga de promover el lenguaje de programación Python principalmente mediante la formación que ofrecen para todas aquellas personas que desean mejorar sus habilidades y conocimientos sobre este lenguaje y las tecnologías que éste lleva asociadas.

Dentro de la oferta de certificaciones que Python Institute pone a nuestro alcance podemos encontrar PCEP, PCAP, PCPP y CEPP. En la siguiente figura podemos ver un esquema general donde cada una de estas certificaciones está ordenada según el nivel de conocimientos y/o aptitudes que acredita la obtención de cada una de estas.

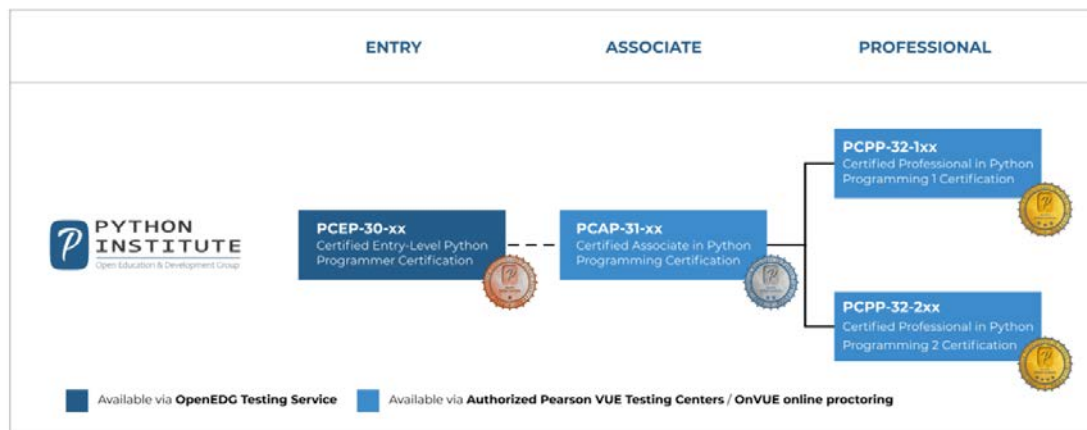


Figura 4.1. Esquema visual de la oferta de certificaciones en Python ofrecidas por Python Institute[6]

A continuación, hablaremos brevemente sobre el temario que cada una de estas certificaciones cubre y las aptitudes que acreditaremos en el supuesto caso de estar en posesión de alguno de estos títulos.

- | **PCEP** - *Certified Entry-Level Python Programmer*. Esta certificación acredita que el individuo está familiarizado con los conceptos básicos de la programación tales como los tipos de datos, funciones, condicionales, bucles, así como la familiarización con la sintaxis de python, su semántica y el proceso de desarrollo general.
- | **PCAP** - *Certified Associate in Python Programming*. Esta certificación establece unos conocimientos más sólidos en el individuo, tanto en el lenguaje de programación de python en particular como con los conocimientos fundamentales en la programación orientada a objetos, constituyendo así un gran primer paso hacia la especialización profesional.
- | **PCPP1** - *Certified Professional in Python Programming 1*. Esta certificación acredita tanto unos conocimientos relativamente avanzados en python como una cierta experiencia en las siguientes áreas: programación orientada a objetos avanzada, programación de GUIs, buenas prácticas y convenciones PEP, proceso de archivos de texto, APIs, etc.
- | **PCPP2** - *Certified Professional in Python Programming 2*. Esta certificación acredita que el individuo posee amplios conocimientos y habilidades en programación en python así como una experiencia importante en las tecnologías asociadas a este, tales como creación y distribución de paquetes, diseño de patrones, IPC, programación de redes y base de datos (MySQL). En definitiva, obteniendo esta certificación demostraremos ser profesionales con muy sólidos conocimientos en Python, preparados para abordar casi cualquier tarea que se nos proponga con un alto porcentaje de éxito.
- | **CEPP** - *Certified Expert in Python Programming*. Finalmente, aquellos estudiantes que superen los exámenes PCAP-31-xx, PCPP-32-1-xx, and PCPP-32-2-xx, son reconocidos con el título de **expertos programadores en python**, constituyendo, ahora sí el “más alto nivel” de reconocimiento tanto en python como en las tecnologías asociadas.

5. ÍNDICE DE CONTENIDOS PCAP

Puesto que los exámenes sufren de continua renovación, es normal que el temario que cubren se vea algo alterado, o al menos modificado el peso en porcentaje de cada uno de los temas. A continuación, veremos una comparativa del temario del PCAP-31-02 frente a la nueva actualización del PCAP-31-03 que entró en vigor el pasado 17 de septiembre de 2020.

5.1 Temario PCAP-31-02 (PVTCS)

Exam block #1: Control and Evaluations (25%)

- | basic concepts: interpreting and the interpreter, compilation and the compiler, language elements, lexis, syntax and semantics, Python keywords, instructions, indenting
- | literals: Boolean, integer, floating-point numbers, scientific notation, strings
- | operators: unary and binary, priorities and binding
- | numeric operators: `** * / % // + -`
- | bitwise operators: `~ & ^ | << >>`
- | string operators: `* +`
- | Boolean operators: not and or
- | relational operators (`== != > >= < <=`), building complex Boolean expressions
- | assignments and shortcut operators
- | accuracy of floating-point numbers
- | basic input and output: `input()`, `print()`, `int()`, `float()`, `str()` functions
- | formatting `print()` output with `end=` and `sep=` arguments
- | conditional statements: if, if-else, if-elif, if-elif-else
- | the pass instruction

- | simple lists: constructing vectors, indexing and slicing, the len() function
- | simple strings: constructing, assigning, indexing, slicing comparing, immutability
- | building loops: while, for, range(), in, iterating through sequences
- | expanding loops: while-else, for-else, nesting loops and conditional statements
- | controlling loop execution: break, continue

Exam Block #2: Data Aggregates (25%)

- | strings in detail: ASCII, UNICODE, UTF-8, immutability, escaping using the \ character, quotes and apostrophes inside strings, multiline strings, copying vs. cloning, advanced slicing, string vs. string, string vs. non-string, basic string methods (upper(), lower(), isxxx(), capitalize(), split(), join(), etc.) and functions (len(), chr(), ord()), escape characters
- | lists in detail: indexing, slicing, basic methods (append(), insert(), index()) and functions (len(), sorted(), etc.), del instruction, iterating lists with the for loop, initializing, in and not in operators, list comprehension, copying and cloning
- | lists in lists: matrices and cubes
- | tuples: indexing, slicing, building, immutability
- | tuples vs. lists: similarities and differences, lists inside tuples and tuples inside lists
- | dictionaries: building, indexing, adding and removing keys, iterating through dictionaries as well as their keys and values, checking key existence, keys(), items() and values() methods

Exam block #3: Functions and Modules (25%)

- | defining and invoking your own functions and generators
- | return and yield keywords, returning results, the None keyword, recursion

- | parameters vs. arguments, positional keyword and mixed argument passing, default parameter values
- | converting generator objects into lists using the list() function
- | name scopes, name hiding (shadowing), the global keyword
- | lambda functions, defining and using
- | map(), filter(), reduce(), reversed(), sorted() functions and the sort() method
- | the if operator
- | import directives, qualifying entities with module names, initializing modules
- | writing and using modules, the __name__ variable
- | pyc file creation and usage
- | constructing and distributing packages, packages vs. directories, the role of the __init__.py file
- | hiding module entities
- | Python hashbangs, using multiline strings as module documentation

Exam block #4: Classes, Objects, and Exceptions (25%)

- | defining your own classes, superclasses, subclasses, inheritance, searching for missing class components, creating objects
- | class attributes: class variables and instance variables, defining, adding and removing attributes, explicit constructor invocation
- | class methods: defining and using, the self parameter meaning and usage
- | inheritance and overriding, finding class/object components
- | single inheritance vs. multiple inheritance
- | name mangling
- | invoking methods, passing and using the self argument/parameter
- | the __init__ method
- | the role of the __str__ method

- | introspection: `__dict__`, `__name__`, `__module__`, `__bases__` properties, examining class/object structure
- | writing and using constructors
- | `hasattr()`, `type()`, `issubclass()`, `isinstance()`, `super()` functions
- | using predefined exceptions and defining your own ones
- | the try-except-else-finally block, the raise statement, the except-as variant
- | exceptions hierarchy, assigning more than one exception to one except branch
- | adding your own exceptions to an existing hierarchy
- | assertions
- | the anatomy of an exception object
- | input/output basics: opening files with the `open()` function, stream objects, binary vs. text files, newline character translation, reading and writing files, bytearray objects
- | `read()`, `readinto()`, `readline()`, `write()`, `close()` methods

5.2 Temario PCAP-31-03 (PVTCS)

Exam block #1: Modules and Packages (12%)

- | import variants; advanced qualifying for nested modules
- | `dir()`; `sys.path` variable
- | math: `ceil()`, `floor()`, `trunc()`, `factorial()`, `hypot()`, `sqrt()`; random: `random()`, `seed()`, `choice()`, `sample()`
- | platform: `platform()`, `machine()`, `processor()`, `system()`, `version()`, `python_implementation()`, `python_version_tuple()`
- | `idea`, `__pycache__`, `__name__`, public variables, `__init__.py`
- | searching for modules/packages; nested packages vs directory tree

Exam block #2: Exceptions (14%)

- | `except`, `except:-except`; `except:-else:`, `except (e1,e2)`

- | the hierarchy of exceptions
- | raise, raise ex, assert
- | event classes, except E as e, arg property
- | self-defined exceptions, defining and using

Exam block #3: Strings (18%)

- | ASCII, UNICODE, UTF-8, codepoints, escape sequences
- | ord(), chr(), literals
- | indexing, slicing, immutability
- | iterating through,
- | concatenating, multiplying, comparing (against strings and numbers)
- | in, not in
- | .isxxx(), .join(), .split()
- | .sort(), sorted(), .index(), .find(), .rfind()

Exam block #4: Object-Oriented Programming (34%)

- | ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass
- | instance vs class variables: declaring, initializing
- | __dict__ property (objects vs classes)
- | private components (instance vs classes), name mangling
- | methods: declaring, using, self parameter
- | introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties
- | inheritance: single, multiple, isinstance(), overriding, not is and is operators
- | inheritance: single, multiple, isinstance(), overriding, not is and is operators
- | constructors: declaring and invoking

- | polymorphism
- | `__name__`, `__module__`, `__bases__` properties, `__str__()` method
- | multiple inheritance, diamonds

Exam block #5: Miscellaneous (List Comprehensions, Lambdas, Closures, and I/O Operations) (22%)

- | list comprehension: if operator, using list comprehensions
- | lambdas: defining and using lambdas, self-defined functions taking lambda as arguments; `map()`, `filter()`;
- | closures: meaning, defining, and using closures
- | I/O Operations: I/O modes, predefined streams, handles; text/binary modes `open()`, `errno` and its values; `close()` `.read()`, `.write()`, `.readline()`; `readlines()` (along with `bytearray()`)

En la siguiente tabla encontramos un resumen de la variación de los porcentajes de peso del temario del PCAP-3102 frente al PCAP-31-03.

No es un cambio muy significativo, pero merece la pena al menos tenerlo en cuenta para saber repartir inteligentemente nuestro esfuerzo.

PCAP-31-02 (PVTCS)		PCAP-31-03 (PVTCS)	
Control and Evaluations	25%	Modules and packages	12%
Data aggregates	25%	Exceptions	14%
Functions and modules	25%	Strings	18%
Classes, objects and exceptions	25%	Object oriented programming	34%
-	-	Miscellaneous (lists comprehensions, lambdas, closures and I/O operations)	22%

Figura 1.6. Comparativa del temario entre los exámenes PCAP-31-02 y PCAP-31-03

6. RECURSOS. OPENEDG Y CISCO NETWORKING ACADEMY

Como mencionamos en puntos anteriores, dentro del ecosistema de instituciones que hacen posible la existencia de las certificaciones de Python, en el apartado educativo, tanto de OpenEDG como Cisco Networking Academy ponen a nuestro alcance material de estudio de gran calidad a través de la herramienta de estudio “edube”.

Si comparamos el material de estudio que al que podemos acceder por medio de OpenEDG o por medio de Cisco, nos daremos cuenta de que son prácticamente idénticos con la diferencia de que Cisco ofrece la posibilidad de escoger el curso en español. Otra diferencia a destacar, es que, en el caso de los cursos proporcionados a través de la web de Cisco, también existe la posibilidad de tener un instructor.

Esta variante suele tener costes añadidos a menos que formen parte de algún programa de becas como es el caso de las “Becas Digitaliza” ofrecidas anualmente por esta institución, donde no solamente ofrecen la posibilidad de recibir este y otros con instructor, sino que también proveen de recursos de aprendizaje extra, acceso a la bolsa de empleo de partners y clientes de Cisco o incluso cupones descuento para realización de exámenes de certificaciones Cisco [7]. Por lo tanto, es muy recomendable también mirarlas.

La primera opción, por tanto, sería dirigirnos a la web (<https://edube.org/login>), crearnos un usuario, y una vez dentro buscar los cursos **“Python Essentials 1 (Basics, v.2.0)”** y también el **“Python Essentials 2 (Intermediate, v.2.0)”** y pulsando simplemente en “Enroll” ya estaremos inscritos y preparados para comenzar el curso. La siguiente imagen muestra el menú de inicio de la plataforma Edube con los dos cursos de Python mencionados.

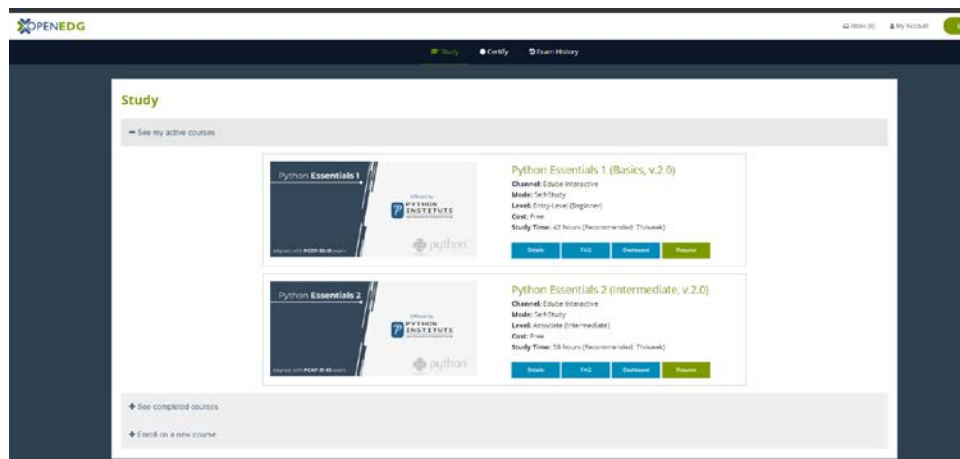


Figura 6.1. Menú de inicio de la plataforma Edube

Por otro lado, también podemos acceder al material (en español) a través de la web de Cisco (<https://www.netacad.com/es>). De la misma manera, nos crearemos un usuario y buscaremos dentro del catálogo de cursos **“PCAP - Programming Essentials in Python Español”**. Veremos que, en este caso, este curso incluye tanto el nivel básico como el intermedio. Además, Cisco añade algunos apartados iniciales como los “Términos y condiciones”, una encuesta inicial y también un apartado relacionado con *preguntas frecuentes*, dónde podremos encontrar respuesta a casi la mayoría de dudas que pudieran surgirnos. A continuación, también una imagen de cómo sería el menú principal del curso.

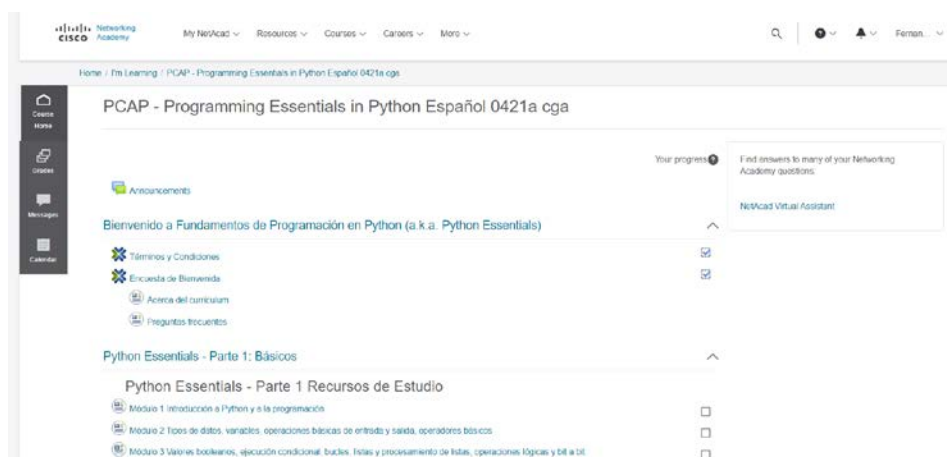


Figura 6.2. Menú principal de la plataforma de estudio del curso de Python en español a través de Cisco Networking Academy

7. OBJETIVOS DE LOS CURSOS

Si bien, los cursos de los que hemos hablados están orientados a cubrir todo el temario mencionado en el punto “Índice de contenidos PCAP”, podemos condensar toda esta información en los siguientes puntos

- | Fundamentos de la programación, cómo funciona un ordenador, como se ejecutan los programas, como se definen y construyen los lenguajes de programación, diferencia entre lenguajes de programación interpretados y compilados, cómo se posiciona python entre los otros lenguajes y las diferencias entre sus versiones 2 y 3.
- | Métodos básicos de formateo y devolución de datos ofrecidos por Python. Principales tipos de datos y operadores numéricos, sus relaciones y dependencias. El concepto de variables y las convenciones de nombramiento de las mismas. Las reglas principales que rigen la construcción de expresiones y el tratamiento del input de datos.
- | Valores booleanos para comparar diferentes valores y controlar los flujos de ejecución utilizando las sentencias “if” e “if-else”. Utilización de bucles (while y for) y como controlar su comportamiento mediante las instrucciones “break” y “continue”. La diferencia entre las operaciones lógicas y las operaciones de comparación bit a bit. El concepto de listas y procesamiento de las mismas mediante las iteraciones con bucles for y el troceado. Concepto de arrays multidimensionales.
- | Definición y uso de funciones y sus características. Como tratar los inputs de estas funciones y los métodos de predefinir valores en las mismas. Retorno de resultados. Tuplas y diccionarios.
- | Módulos de Python. Razones para su existencia, cuál es su función. Cómo importarlos de diferentes maneras. Módulos más conocidos o estándar en Python. Generación de paquetes. Excepciones (sentencias “try-except”) y sus aplicaciones. Instrucción “raise”. Strings y sus principales métodos y su comparativa con listas.
- | Fundamentos de POO (Programación Orientada a Objetos) y diferencia con la clásica programación procedimental. Características principales de la POO; herencia, abstracción, encapsulación, polimorfismo, etc. Generadores y procesamiento básico de archivos (create, read, write).

8. EXAMEN PCAP

En la web de *Python Institute* podemos encontrar toda la información que necesitemos sobre el examen. Algunos datos importantes sobre este examen son:

- Prerrequisitos: Ninguno
- Modalidades de examen: Centros certificadores u online a través de Pearson onVUE Online Proctoring
- Duración del examen: 65 minutos (examen)
- Número de preguntas: 40 preguntas
- Calificación necesaria para pasar el examen: 70%
- Tipo de preguntas: Preguntas de opción múltiple con una o varias respuestas

8.1 Precio del PCAP y descuentos disponibles

Particularmente, uno de los datos que suele interesar bastante a los estudiantes, y con razón, es el precio que tiene la realización del examen.

Existen 3 opciones o precios, según la situación en la que nos encontremos, o mejor dicho, según hayamos o no superado con éxito los cursos de los cuales hemos hablado en el punto 0.

- | La primera opción (**Full exam fee**) contempla la situación en la que decidamos presentarnos al examen directamente sin haber superado ningún curso previo y, por lo tanto, sin ningún tipo de descuento. En el caso del PCAP, el precio es de **295 USD**.
- | Por otra parte, la segunda opción (**Open Enrollment**) contempla el haber superado todos los módulos del curso “Python Essentials 2 (Intermediate, v.2.0)” de OpenEDG (para el cual necesitaremos cursar el básico también, aunque los criterios de nota media mínima no se aplican a este) con una nota media **mayor o igual a 7** (tanto de los tests de cada módulo como del examen final). En este supuesto caso, se nos otorgará un descuento del 50% del precio del examen, quedando finalmente este en **147.50 USD**.

Por último, también tendremos a nuestro alcance un descuento si realizamos los cursos a través del portal de Cisco Networking Academy. En el apartado de preguntas frecuentes se nos explica que los requisitos para tener acceso a este descuento son muy similares a los explicados en OpenEDG.

- En primer lugar, haber superado la Evaluación Parcial 1 (obteniendo al menos 21/30 puntos) y la Evaluación Parcial 2 (obteniendo al menos 21/30 puntos).
- Haber superado el Examen final (obteniendo al menos 35/50 puntos).
- Completar una encuesta de satisfacción final.

Como vemos, los requisitos son prácticamente los mismos, salvo por la encuesta, pero sin embargo, el descuento sí que es distinto, no por mucho, es verdad, pero en este caso se otorga a los estudiantes un descuento del 51% sobre el total, con lo cual, finalmente, el precio se queda en unos **144.55 USD**.

9. ACREDITACIÓN DIGITAL

Finalmente, para concluir esa lección, hablamos brevemente de la certificación digital a la que tenemos acceso una vez superado el examen del PCAP y la plataforma mediante la cual podemos compartir nuestros logros con nuestra red de contactos.

Si bien es cierto que actualmente sigue siendo habitual recibir una certificación en formato papel al superar el examen de cualquier certificación (incluso al superar el PCAP recibiremos la nuestra) cada vez es más común la digitalización de estos certificados con el fin de hacerlos mucho más seguros ante la pérdida o falsificación de los mismos a la vez que facilita la exposición de nuestros logros en nuestros perfiles profesionales de plataformas de búsqueda de empleo como LinkedIn o Infojobs otorgándonos una mayor visibilidad.

Este reconocimiento digital de nuestras aptitudes se logra a través de la plataforma **Credly's Acclaim**, y el proceso que seguiríamos para obtener finalmente esta certificación sería el siguiente:

- | Superar el/los exámenes de PCAP/PCPP1 (Para poder obtener el certificado digital del PCPP1 es necesario tener el PCAP también).
- | Posteriormente, recibiremos un email de *Acclaim* donde se nos invitará a solicitar nuestro certificado en su plataforma.
- | Para ello, será necesario crear una cuenta de Acclaim o logearse si ya tuviésemos una.
- | Aceptar el certificado
- | Compartirlo en nuestros portales de empleo.

10. BIBLIOGRAFÍA

- [1] Wikipedia. (2022). Anexo: Lenguajes de programación. Available: https://es.wikipedia.org/wiki/Anexo:Lenguajes_de_programaci%C3%B3n
- [2] Wikipedia. (2022). Anexo: Cronología de los lenguajes de programación. Available: https://es.wikipedia.org/wiki/Anexo:Cronolog%C3%ADa_de_los_lenguajes_de_programaci%C3%B3n
- [3] T. S. BV. TIOBE Index for January 2022. Available: <https://www.tiobe.com/tiobe-index/>
- [4] PYPL. PYPL PopularitY of Programming Language. Available: <https://pypl.github.io/PYPL.html>
- [5] T. S. BV. TIOBE Programming Community Index Definition. Available: <https://www.tiobe.com/tiobe-index/programming-languages-definition/>
- [6] P. Institute. Python Institute Certification Roadmap. Available: <https://pythoninstitute.org/certification/>
- [7] C. N. Academy. Becas Digitaliza. Available: <https://gblogs.cisco.com/es/2022/01/vuelven-las-becas-rethinking-digital-de-formacion-en-competencias-digitales/>